HOW SECOPS IMPROVES IT SECURITY THROUGH A SHIFT-LEFT APPROACH

Most of the time when we talk about SecOps, we are really implying DevSecOps. If you aren't familiar with the DevSecOps concept, I suggest reading Rick Bosworth's article on What is DevSecOps? DevSecOps Explained.

However, not all organizations have completely transitioned over to DevOps. In today's technology environments, there is still a mix of modern and legacy technologies. In most large organizations you'll find a mix of cloud technologies, traditional client/server applications, and sometimes legacy mainframe (or mainframe like) systems. The architecture for these systems is oftentimes based on a model where security was applied after the initial configuration. Frequently, operational ability trumps security. Every IT leader I know has been faced with making the decision on whether to meet a project deadline for rollout by "cutting the security corner" or to delay system delivery to ensure sufficient security has been applied. As the saying goes, if I had a dollar for every time I heard, "let's go ahead and launch, we can finish up security after we go live" then I'd be writing this article from my private yacht.

SecOps Defined

Donovan Brown, Microsoft DevOps PM defines DevOps by saying "DevOps is the union of people, processes and products to enable continuous delivery of value to end users."

If we take this definition one step further, we can define SecOps as:

> *"The union of existing rapid software development and infrastructure processes with security functions that result in reduced risk to the organization while enabling continuous delivery to end users.*

In short, SecOps is the integration of security throughout software development and infrastructure deployment, rather than the addition of security as the last step before deployment.

Key Concepts

Change in Culture

The infamous Peter Drucker one said that *"culture eats strategy for breakfast."* Ultimately, SecOps is more of a change in organizational culture than anything else. Over the years, various IT departments have siloed themselves from one another. This was largely done to address the complexity that is inherent in large IT environments. The main objective was to limit change to ensure system uptime and reliability.

With the transition to digital business, rapid development and deployment of new technologies can be a significant competitive advantage. In order to remain competitive organizations must now provide reliability and embrace continual change. This means that we can't throw things "over the wall" to security to deal with on their own. Nor can security say "that's an operations issue, we identified the vulnerability." The SecOps model breaks down those walls and requires that security and operations work together and share ownership of issues.

Rather than approaching your transition to SecOps as a traditional project, approach it as a behavioral change. Changing the culture will go a long way towards defining what business processes enable a move towards SecOps.

Shared Ownership

One of the key concepts in DevOps is the removal of the silo between development and operations. In DevOps, development and operations both share ownership (and responsibility) for delivery of new code. Development is not allowed to get credit for completion of work at the handoff to operations. DevOps requires that code must make it all the way to production before either team can claim work as completed.

Security must now also be included in the ultimate ownership of delivery. In SecOps security tests are performed as part of (and in line with) other testing such as performance and reliability. Furthermore, the security team must also share accountability for the delivery of code all the way to production.

The shared responsibility model continues after software enters production. If a security vulnerability is discovered that must be addressed by the dev or operations team, all three teams are held accountable for addressing the issue as quickly as possible.

Automation

Haresh Sippy is quoted as saying: *"Automation is cost cutting by tightening the corners and not cutting them."* This concept directly applies to the concept of SecOps. By adding more frequent testing to your development or deployment processes you are definitely introducing more work. More work means increased cost to move systems into production. The automation of as much security testing as possible is the key to ensuring that the increase to personnel costs are kept to a minimum.

I can personally attest to the fact that human error is usually the root cause of most data security issues. According to Gartner, 95% of cloud data breaches over the next few years will due to misconfiguration. In cloud environments where infrastructure and configuration can be deployed as code, we can automate many of the previous, manual configuration tasks. This increased level of automation can creates consistency in system configuration which makes security and performance much more predictable.

Process AND Products

Every PMO will tell you that your project management process is more important than your project management system. Additionally, every CISO will tell you that security is a process not a product. However, both project management and security ultimately require products in order to deliver their respective processes to the organization. SecOps is no different. In order to effectively leverage SecOps practices, you'll need tools to help you automate security testing. With multi-cloud environments the need for proper management tools only becomes more important. Be sure to include product evaluation and testing as part of your SecOps implementation efforts. Look for products that can give you the ability to identify and prioritize security tasks based on organizational (business) metrics. Also important is the ability to compare security policies across multiple cloud providers. With SecOps there will be more smaller tasks to track than there was previously. Make sure your SecOps tools help you track the higher quantity of tasks.

A Shift Left

Ajoy Kumar provides a great explanation of the shift left concept in his blog entitled "5 Best Practices Building Security as Code." In his post, he suggests the following:

> *"*
>
> *Instead of approaching security as an add-on process at the end of the release, or as periodic security scanning of production environments, start with security at the beginning of an application release. Represent it as "code" and bake it in.*

While his example is specific to development and integration into the DevOps process, the concept still works in the boarder SecOps concept. Let's take a look at one example:

A "Shift Left" Scenario Example

Think about your traditional enterprise application deployment. The first step is typically the requisition of new servers based on the application requirements and estimated user workload.

Next, the servers are built (either physical or virtual) from existing template installations. Then, the bits for the application are made available to the server and the application is installed. Hopefully, the systems administrators (or vendor) installing the application remembers all ~200 questions of your security framework and meets every requirement. Last, (if it happens at all) security testing is performed on the "as built" installation of the application before it enters production. At this point the security team identifies any issues and (at least to the sysadmin) delays the application from entering production. With the two teams feuding, the CIO (or senior director) must now make a decision. Do I delay the project for security, or do I release the application, meet my deadline, and have the security team do cleanup later?

Now, let's revisit this scenario and borrow the "shift left" concept from DevSecOps. Before servers are even requisitioned for the project, security and operations have co-created server templates that meet security baselines and are version controlled. Because they share responsibility for image creation, security was able to offer input into the server builds (and assist when needed) and operations was able to offer feedback on how to implement security controls. Furthermore, this work was completed before either team was under a deadline to deliver an application. Now, when the server requisition comes through, the organization is already starting with a "known secure" configuration. When the bits for the application are made available for installation, both security and operations play a part in the installation. Security issues are identified and remediated during the installation process. Security may even have the opportunity to perform some testing as various parts of the installation take place. Now, when the installation is complete, security has had a role throughout the installation. Most importantly, security has an understanding of how the application works and has a much higher (hopefully) comfort level with the application. Final security testing can be completed much faster since most of the work was done along the way. The CIO is much happier with both teams since he is not forced into a no-win decision.

Closing Thoughts

Whether you take the DevSecOps approach or the SecOps approach I have defined here, most organizations can benefit from a revised approach to security. By integrating security practices at the beginning, and throughout system implementation efforts, organizations can reduce silos, change culture, reduce risk and ultimately deliver value much faster to customers.